Model Order Reduction for Large Scale Engineering Models Developed in ANSYS

E. B. Rudnyi, J. G. Korvink

IMTEK, Universität Freiburg, Georges-Köhler-Allee, 103
D-79110, Freiburg, Germany, {rudnyi, korvink}@imtek.de
http://www.imtek.uni-freiburg.de/simulation/

Abstract. The software mor4ansys that allows engineers employ the modern model reduction technique to the finite element models developed in ANSYS is presented. We focus on how one extracts the required information from ANSYS and makes the model reduction implementation not dependent on a particular sparse solver in C++. We discuss the computational cost and give examples related to structural mechanics and thermal finite element models.

1 Introduction

Model order reduction of linear large-scale dynamic systems is already quite an established area [1]. In many papers (see references in [2]), the advantages of model reduction have been demonstrated for variety of scientific and engineering applications. In the present work, we focus on how engineers can combine this technique with existing commercial finite element software in order to

- Speed up a transient or harmonic analysis,
- Generate automatically compact models for system-level simulation,
- Incorporate finite element packages during the design phase.

Model reduction is conventionally applied to a large-scale dynamic system of the first order as follows

$$\begin{aligned} E\dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} \end{aligned} \tag{1}$$

where A and E are the system matrices, B is the input matrix, C is the output matrix. The aim of model reduction is to generate a low-dimensional approximation in the similar form

$$\begin{aligned} E_r\dot{\boldsymbol{z}} &= A_r\boldsymbol{z} + B_r\boldsymbol{u} \\ \boldsymbol{y} &= C_r\boldsymbol{z} \end{aligned} \tag{2}$$

that describes well the dependence of the output vector $\boldsymbol{y}$ on the input vector $\boldsymbol{u}$ and at the same time the dimension of the reduced state vector $\boldsymbol{z}$ is much less than the dimension of the original state vector $\boldsymbol{x}$.

The finite element package produces a system of ordinary differential equations after the discretization in space the partial differential equation describing the user model. At this stage, it is possible to apply directly the modern model reduction methods [1]. However, the extraction of the system matrices from a commercial package happens not to be an ordinary problem and we share our experience on how it can be done with ANSYS [3].

The system matrices are high dimensional and sparse. As a result, the implementation of a model reduction algorithm usually depends on a particular sparse solver and a storage scheme for sparse matrices. We present C++ interface that allows us to isolate the model reduction and sparse solvers completely for the negligible overhead.

Finally we analyse the computation cost and give the performance results for a few ANSYS models. The comparison of the accuracy of reduced models in respect to the original ANSYS models is given elsewhere [4].

2 mor4ansys

The developed software comprises the two almost independent modules. The first reads binary ANSYS files and assembles a dynamics system in the form of Eq (1) for the first order systems or

$$\begin{aligned} M\ddot{\boldsymbol{x}} + E\dot{\boldsymbol{x}} + K\boldsymbol{x} &= B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} \end{aligned} \tag{3}$$

for the second order systems, where M, E and K are the three system matrices for the second-order systems. The second module applies the model reduction algorithm to Eq (1) or (3), that is, it finds a low-dimensional subspace V

$$\boldsymbol{x} = V\boldsymbol{z} + \varepsilon \tag{4}$$

such that it allows us to reproduce the transient behaviour of the original state vector with required accuracy.

After that, the original equations are projected to the subspace found, for example for Eq (2) we have $E_r = V^T E V$, $A_r = V^T A V$, $B_r = V^T B$, $C_r = CV$.

The software can also read as well as write the matrices for the original system in the Matrix Market format [5].

2.1 Interfacing ANSYS

The development of the first module happened to be rather difficult because the most users of a commercial finite element package do not want to extract the dynamics system in the form of Eq (1) or (3) and, as a result, this is not a trivial operation.

ANSYS is a huge package and its behaviour is not completely consistent. For example, the information described below is not applicable for the fluid dynamics module FLOTRAN.

Our software uses a binary `EMAT` file with the element matrices in order to assemble the global system matrices. The file format is documented and ANSYS supplies a library of Fortran subroutines to work with it [6]. An example of how one can use them

can be found in the Mathlink application that allows us to read binary ANSYS files into Mathematica [7]. ANSYS has a special command, partial solve, PSOLVE, when one can estimate element matrices for given a state vector without going through the real solution stage. This allows us to generate an EMAT file quite fast for a given model. However, it was necessary to overcome the next problems:

- The EMAT file does not contain the information about neither Dirichlet nor equation constraints. They should be extracted separately.
- The EMAT file has a contribution to the load vector from element matrices only. If nodal forces or acceleration is used to apply the load, this information should be also extracted individually.
- It is necessary to assemble the global matrices from the element ones.

During the solution phase, ANSYS can write a binary FULL file with the assembled system matrices. When we have started the development with ANSYS 5.7, this file did not contain the load vector (input matrix). Since then there have been many changes. Since ANSYS 6.0 the FULL file can keep all the original matrices, the load vector, the Dirichlet and equation constraints. The newest version ANSYS 8.0 allows us to make the assembly only and write the FULL file without a real solution phase (equivalent to partial solution with EMAT). One can also now dump the information from the FULL file in the Harwell-Boeing matrix format. Hence, since ANSYS 8.0 it is possible to use the FULL file efficiently. However, depending on the analysis type the FULL file may contain not the original stiffness matrix but rather the linear combination of system matrices instead. Thus, it is still fun.

In the current version of mor4ansys, the EMAT files is employed as the main source to build Eq (1) or (3). An additional information is written in the form of text files, which are generated by means of ANSYS macros developed by us: the Dirichlet and equation constraints and nodal forces. The FULL file can be used to extract the load vector when otherwise this is difficult, for example, when the acceleration load is used.

ANSYS cannot write a few different load vectors into the FULL and EMAT files. When multiple-input is to be preserved in Eq (1) or (3), a use should for each input:

- Delete previously applied load,
- Apply a new load,
- Generate matrices.

In order to ease this process, the second strategy is also allowed when a user does not have to delete the previous load. In this case, each new load vector contains all the previous vectors and mor4ansys correct them at the end of the first phase.

2.2 Running Model Reduction Algorithm

The Krylov subspaces allow us to obtain a low-dimensional subspace in Eq (4) with excellent approximating properties and at the same time by means of a very efficient computational way [8][9]. The current version of mor4ansys implements the block Arnoldi algorithm [8].

Each step of an iterative Krylov subspace algorithm requires us to compute a matrix-vector product, for example, for the first-order system

$$A^{-1}E\boldsymbol{h} \tag{5}$$

where $\boldsymbol{h}$ is some vector. The system matrices are high-dimensional and sparse and one cannot afford to compute A^{-1} explicitly. The only feasible solution is to solve a linear system of equations for each step as follows

$$A\boldsymbol{g} = E\boldsymbol{h} \ . \tag{6}$$

This constitutes the main computational cost up to the order of the reduced system 30. Later on, the additional cost associated with the orthogonolization process can be also observed.

There are many sparse solvers as well as many storage schemes for sparse matrices. Our goal was to implement a model reduction algorithm in a way that does not depend on a particular solver. In addition, we wanted to change solvers at the run-time, that is, to allow for the run-time polymorphism. As a result, we have chosen the virtual function mechanism, as its overhead is negligible in our case when the operations by themselves are quite computationally intensive.

The basic operations, which in our view should enough to support the Krylov-subspace based model reduction algorithms, are expressed as follows

```
class LinearSolver {
public:
virtual MatrixHandle SetMatrix(SparseMatrix &mat) = 0;
virtual MatrixHandle SumMatrices(MatrixHandle mat1,
MatrixHandle mat2, const double &alpha) = 0;
virtual void ClearMatrix(MatrixHandle mat) = 0;
virtual void MulMatrixByVec(MatrixHandle mat, double
*in, double *out) = 0;
virtual InverseHandle PrepareInverse(MatrixHandle mat,
bool ClearMatrix, const string &param) = 0;
virtual void ClearInverse(InverseHandle L) = 0;
virtual void MulInverseByVec(InverseHandle L, double
*in, double *out) = 0;
virtual bool IsSymmetric(MatrixHandle mat) = 0;
virtual void clear() {}
virtual ~LinearSolver() {clear();}
};
```

The class is written in terms of relatively low-level functions, as the goal was to cover many different scenarios. The vectors are represented by continuous memory, as they are dense in the case of the Krylov subspaces.

The function `SetMatrix` takes a sparse matrix from the first module and converts it to the form required by the solver. It returns a low-weight `MatrixHandle` object that is currently represented as a pointer to void. It is the user responsibility to free memory for the matrix by calling `ClearMatrix`.

The function `PrepareInverse` computes either a factor or preconditioner. It can delete the original matrix if necessary. The low-weight `InverseHandle` object, a

pointer to void in our implementation, is responsible to point to the data structure that keeps all relevant information for the solution phase.

The function `clear` and destructor is responsible to delete the internal data structures provided they were needed.

At present, the direct solvers from the TAUCS [10] and UMFPACK [11] libraries are supported. The ATALS library [12] has been used to generate the optimized BLAS. We have found that for many ANSYS models up to 500 000 degrees of freedom the modern direct solvers are quite competitive as the matrix factor fits within 4 Gb of RAM. This allows us to reuse the factor to generate V and achieve good performance.

3 Computational Cost of Model Reduction

We have observed that for many ANSYS models the order of the reduced system 30 is enough to accurately represent the original high-dimensional system [4]. Hence, for simplicity we limit the analysis of the computational cost to this case.

The simulation time of the reduced system comprising 30 equations is very small and we can almost neglect it. Thereafter, in the case when several simulations with different input functions are necessary (the system-level simulation case), the advantage of model reduction is out of question.

Yet, during the design phase the reduced model should be generated each time when a user changes the geometry or material properties in the original model. In this case, a reduced model might be used just once. Nevertheless, the model reduction time can be smaller than the simulation time of the original system even in this case. Let us consider this case below.

Let us assume that a direct solver is applicable and the dimension of 30 for the reduced system is enough. Then the model reduction time is equal to the sum of factoring of A in Eq (5) and for the 30 back substitution steps in Eq (6). Table 1 presents computational times for seven ANSYS models where the system matrices are symmetric and positive definite. The first four rows correspond to the thermal simulation [13] and the last three to the structural mechanics of a bonded wire [14].

Each case is specified by its dimension and the number of non zero elements in the stiffness matrix. Then he time of a stationary solution in ANSYS is given as a reference point. Note that the real simulation time in ANSYS required for the stationary solution is bigger than in Table 1 as it includes reading/writing files as well as some other operations. After that go the time to factor a matrix by means of the multifrontal solver from the TAUCS library [10] and the time to generate the first 30 vectors. The latter is dominated by the solution of Eq (6) by means of back substitution. As the difference to generate the first and thirtieth vectors was less than 10-20%, we can say that the orthogonolization cost was relatively small.

Note that the TAUCS multifrontal solver is even faster than the ANSYS solver. The total time to generate a reduced model is about two times more than that for the stationary solution. At the same time, the reduced model can accurately reproduce any transient and harmonic simulation [4] of the original models.

Table 1: Computational times on Sun Ultra-80 with 4 Gb of RAM in seconds

dimension	nnz	stationary solution in ANSYS 7.0	stationary solution in ANSYS 8.0	factoring in TAUCS	generation of the first 30 vectors
4 267	20 861	0.87	0.63	0.31	0.59
11 445	93 781	2.1	2.2	1.3	2.7
20 360	265 113	16	15	12	14
79 171	2 215 638	304	230	190	120
152 943	5 887 290	130	95	91	120
180 597	7 004 750	180	150	120	160
375 801	15 039 875	590	490	410	420

The simulation time of a harmonic analysis is a product of solution time for a complex linear system by the number of frequencies needed. The matrix factor cannot be re-used as the linear system to solve depends on the frequency. The solution time for a complex linear system is about two times more expensive. Hence model reduction allows us to save the simulation time by a factor equal to a number of frequencies at which the harmonic response is required. For example, if it is necessary to estimate the transfer function at ten frequencies, then the model reduction plus the simulation of the reduced system is roughly ten times faster than the simulation of the original system.

For the transient simulation, the situation is more difficult to analyse as this depends on the integration strategy. In principle, it is possible to say that the model reduction time above is equivalent to 30 equally spaced timesteps as in this case the same strategy with the re-use of the matrix factor can be applied. However, in our experience in order to receive accurate integration results for the examples in Table 1, one either needs at least 600 equally spaced timesteps or the use of adaptive integration schemes where the factor re-use in not possible. In both cases, the model reduction plus the simulation of the reduced system was more than ten times faster.

4 Conclusions

We have shown that in the case of linear dynamics system (1) and (3) modern model reduction techniques can speed up finite element transient and harmonic simulation significantly. For nonlinear systems, there are promising theoretical results in the case of polynomial type nonlinearity [15]. Yet, in the nonlinear case in addition to many theoretical problems, it happens that extracting a nonlinear system (1) or (3) from a commercial finite element tool is a challenge by itself.

5 Acknowledgment

ANSYS models of the microthruster and the bonded wire have been made by T. Bechtold and J. Lienemann respectively. Partial funding by the DFG project MST-Compact (KO-1883/6), the Italian research council CNR together with the Italian province of Trento PAT, the European Union (grant EU IST-1999-29047, Micropyros) and an operating grant of the University of Freiburg is gratefully acknowledged.

6 References

[1] A. C. Antoulas, D. C. Sorensen, Approximation of large-scale dynamical systems: An overview, Technical Report, Rice University, Houston, 2001, http://www-ece.rice.edu/~aca/mtns00.pdf.

[2] E. B. Rudnyi, J. G. Korvink, Automatic Model Reduction for Transient Simulation of MEMS-based Devices, 2002, Sensors Update 11 (2000) 3–33.

[3] ANSYS, ANSYS Inc., http://www.ansys.com.

[4] J. G. Korvink, E. B. Rudnyi, Model Order Reduction of MEMS for efficient computer aided design and system simulation, Sixteenth International Symposium on Mathematical Theory of Networks and Systems, Belgium, July 5-9, 2004.

[5] R. F. Boisvert, R. Pozo, K. A. Remington, The Matrix Market Exchange Formats: Initial Design, NISTIR 5935, http://math.nist.gov/MatrixMarket/

[6] Guide to interfacing with Ansys, ANSYS Inc. It is not included with ANSYS, one has to purchase it separately.

[7] E. B. Rudnyi, AnsysRecords and AnsysEmat, IMTEK Mathematica Supplement, http://www.imtek.uni-freiburg.de/simulation/mathematica/IMSweb/. See Add-Ons | IMTEK | Interfaces and files to compile at AddOns/Applications/Imtek/Interfaces/Ansys/

[8] R. W. Freund, Krylov-subspace methods for reduced-order modeling in circuit simulation, Journal of Computational and Applied Mathematics, 123 (2000) 395-421.

[9] Z. J. Bai, Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems, Applied Numerical Mathematics 43 (2002) 9–44.

[10] S. Toledo, D. Chen, V. Rotkin, TAUCS – A library of sparse linear solvers, http://www.tau.ac.il/~stoledo/taucs/

[11] T. A. Davis, UMFPACK, http://www.cise.ufl.edu/research/sparse/umfpack/

[12] Automatically Tuned Linear Algebra Software (ATLAS), http://math-atlas.sourceforge.net/

[13] E. B. Rudnyi, Micropyros Thruster, 2004 http://www.imtek.uni-freiburg.de/simulation/benchmark/

[14] E. B. Rudnyi, J. Lienemann, A. Greiner, and J. G. Korvink, mor4ansys: Generating Compact Models Directly from ANSYS Models. Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show, Nanotech 2004, March 7-11, 2004, Bosten, Massachusetts, USA.

[15] J. R. Phillips, "Projection-based approaches for model reduction of weakly nonlinear, time-varying systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 22, (2003) 171-187, 2003.